APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: OPTIMIZED LOOK-UP TABLE CALCULATIONS IN BLOCK DIAGRAM SOFTWARE

APPLICANT: ROBERT OLSON ABERG

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL298425594US

I hereby certify under 37 CFR §1.10 that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

July 20, 2001
Date of Deposit

Samantha Bell
Signature

Samantha Bell
Typed or Printed Name of Person Signing Certificate

OPTIMIZED LOOK-UP TABLE CALCULATIONS IN BLOCK DIAGRAM SOFTWARE

BACKGROUND

The invention relates generally to graphical block diagram modeling.

Dynamic systems may be modeled, simulated and analyzed on a computer system using graphical block diagram modeling. Graphical block diagram modeling graphically depicts mathematical relationships among a system's inputs, states and outputs, typically for display on a graphical user interface.

In the context of graphical block diagram modeling, blocks are graphical structures that have a defined functionality. Such blocks may be placed in a reference library to define a graphical class. Graphical libraries are similar to system software libraries in that they are a repository of classes. These classes, or more specifically, graphical classes, can be used by multiple users to build custom systems. When a graphical class is used in a model, it is said to be instantiated, i.e., an instance of the graphical class is created for use in the model. Such an instance can also be referred to as a link to the graphical class that resides in the library. Parameters are class member data that are specified when a user constructs a new instance of a class. Instances associated with the class have parameter specification interfaces that allow a user to define these "top-level" parameters. On a graphical user interface (or "GUI"), such parameter specification interfaces take the form of a dialog box with various parameter entry fields. A user can therefore change the top-level parameters of an instance by changing parameter data in a parameter field in the dialog box for that instance.

Prior graphical block-diagram based tools for modeling and simulating dynamic systems include Simulink® from The MathWorks Inc. One type of block supported by such tools is an n-dimensional interpolation block that performs an index search operation and interpolated table lookup. Multiple interpolation blocks with common input signals and breakpoint data sets have to perform identical index search operations.

SUMMARY

In one aspect, the invention provides methods and apparatus, including computer program products, for block diagram modeling. The methods include a method of block

diagram modeling in a data processing system. The method includes: in a first block, receiving a first value indicative of an index into a lookup table; in the first block, generating information indicative of the location of the first value relative to a predefined domain of possible indexed values; in a second block, receiving the information generated by the first block; and using the information received in the second block to determine an output value of a first lookup table.

Particular implementations of the invention may provide one or more of the following advantages.

Unlike prior graphical block diagram approaches, which for multiple table look-up blocks with common input signals and breakpoint data sets repeat identical index search operations for each input value and breakpoint data set specified by a user, the graphical block diagram technique of the present invention allows the result of each index search block calculation to be re-used by every interpolation block semantically connected to that pre-lookup block's outputs. Additionally, in the case of multi-dimensional table interpolations, the pre-lookup index search block reduces the specification of the index search parameters for each breakpoint data set. Thus, the technique provides for a more efficient multi-dimensional interpolation, as it does not require repeating calculations and allows centralized specification of index search parameters.

Other features and advantages of the invention will be apparent from the following detailed description and from the claims.

DESCRIPTION OF DRAWINGS

FIGS. 1A and 1B are prior art, graphical block diagramming software generated block diagrams that include look-up blocks for 1-dimensional and 2-dimensional interpolation, respectively. Figure 1B in particular shows four 2-dimensional interpolation blocks that share common input signals and breakpoint sets for the respective input signals.

FIG. 2 is a block diagram of a system that includes a file server that stores a graphical block diagramming module that includes model editor, simulator and software code generation processes that support pre-lookup index search blocks and interpolation (using pre-lookup index search) blocks, and user systems that access the file server and execute the processes of the graphical block diagram modeling/simulation module for graphical block

diagram model development and execution.

FIGS. 3A-B are exemplary 1-dimensional and 2 dimensional depictions, respectively, of a block diagram that employs the pre-lookup index search block and interpolation (using pre-lookup index search) block of FIG. 2.

FIG. 4 is an exemplary table for 2-dimensional index search and interpolated lookup.

FIG. 5 is an exemplary screen display from a GUI of a client system executing the model editor of the graphical block diagram modeling/simulation module (of FIG. 1) during graphical block diagram model development, illustrating the use of the conventional interpolation look-up block (of FIG. 1B) along side the use of the pre-lookup index search blocks and interpolation (using pre-lookup index search) block.

FIG. 6 is an exemplary GUI screen display of a dialog box for specifying block parameters for the conventional interpolation look-up block shown in FIG. 5.

FIGS. 7A and 7B are exemplary GUI screen displays of respective dialog boxes for specifying block parameters for the pre-lookup index search blocks shown in FIG. 5.

FIG. 8 is an exemplary GUI screen display of a dialog box for specifying block parameters for the interpolation (using pre-lookup index search) block shown in FIG. 5.

FIG. 9 is another view of the GUI screen display of the block diagram of FIG. 5 showing the selection of the interpolation (using pre-lookup index search) block for use by a debugger tool.

FIG. 10 is an exemplary GUI screen display of the debugger tool results showing the input and output values for the selected interpolation (using pre-lookup index search) block during model execution.

Like reference numbers will be used to represent like elements.

DETAILED DESCRIPTION

FIGS. 1A and 1B illustrate conventional interpolation table lookup in the form of a graphical block diagram. Referring to FIG. 1A, a first block diagram 10 includes a one-dimensional interpolation lookup block 12 which receives an input value ("u_1") from a single input port ("In1") 14 and provides an output value ("y_1") to a single output port ("Out1") 16. Referring to FIG. 1B, a second block diagram 18 includes four two-dimensional interpolation lookup blocks 20a, 20b, 20c and 20d. Each of the blocks 20

receives a first input value ("u_1") from a first input port ("In1") 22 and a second input value ("u_2") from a second input port ("In2") 24, and provides an output value to a corresponding output port ("Out1") 26a, ("Out2") 26b, ("Out3") 26c, ("Out4") 26d, respectively.

The manner in which the blocks 12, 20 are defined and executed to perform an index search operation and interpolated table lookup is known in the art. The lookup table blocks 12, 20 implement an array of table data and map the table data to a domain of indexed values ("breakpoints"), which is typically a non-repeating, monotonically increasing set of values. The breakpoints are the values at which the relationship which the table has sampled is evaluated for a given index. The breakpoints define intervals or segments in which the input value(s) may fall. The blocks determine the location of the input value relative to one of the intervals and use that location to interpolate between table data values. It will be appreciated that, in a multi-dimensional configuration such as that shown in FIG. 1B, each of the blocks 20 has to perform its own index search operation for the same set of input values and the same breakpoint sets.

Referring to FIG. 2, a system 30 includes client systems 32a, 32b,...., 32k and a file server 34 each connected to a network 36, e.g., an Ethernet network, the Internet, or other type of network. Alternatively, instead of being connected to a network, the systems 32 and 34 could be interconnected by a system bus such as Fibre Channel. Each of the client systems 32a, 32b, ..., 32k includes a graphical user interface (GUI) 38a, 38b, ..., 38k. The file server 34 is configured with a graphical block diagram modeling and simulation module 40 (hereinafter, simply, "the module"), which is implemented in software. The module 40 includes a model constructor/editor 42, as will be described later. The module further includes a blocks library 44 and a block diagram processing engine 46. As will be explained more fully below, the model editor 42, in conjunction with the library 44, is used by a client system user to construct and display a graphical block diagram model which visually and pictorially represents a dynamic system of interest to that user. The block diagram processing engine 46 includes a compiler/linker 48 to interpret the block diagram generated by the model editor 42, a simulator 50 that executes the interpreted block diagram to produce simulation results and a code generator 51 that converts the interpreted block diagram to executable code (e.g., textual source code, such as C, or firmware source code for execution on an embedded processor), or transmits signals that configures or can be used to configure a

hardware device, for example, a field programmable gate array.

The block library 44 serves a repository for blocks, including a pre-lookup index search block 52 which performs an index search operation and an "interpolation using pre-lookup index search" block (hereafter, simply, "interpolation block") 54, which uses the output of the pre-lookup index search block 52 to perform an interpolated lookup, as will be described.

The system 30 illustrates a remote client access configuration in which the module 40 is installed on a central file server, i.e., the file server 34, and users of the client systems 32 access the module 40 over the network 32. In an alternative configuration, e.g., in an environment in which access to the library 44 is not shared by multiple users, the module 40 could be installed on a single stand-alone or networked computer system for local user access.

Each of the client systems 32a, 32b,..., 32k, includes a respective memory 56a, 56b,..., 56k, for storing all or accessed portions of the module 20, as well as a respective CPU 58a, 58b, ..., 58k for executing the stored portions of the module 40, the GUI 38 and other OS programs (not shown) for controlling system hardware. Although not shown, it will be understood that the systems 32 and 34 can be, in other respects, of a conventional design and architecture. That is, the systems 32 include conventional system I/O peripherals, e.g., display, mouse, keyboard and the like, for enabling user interaction with the system. The file server 34 includes conventional server software and hardware, and thus includes the appropriate storage for storing the software programs of the module 40, along with OS and server application programs, and CPU for executing those programs.

For illustrative purposes, the module 40 will be described within the context of a Simulink® and MATLAB® based simulation environment. Simulink® and MATLAB® are commercial software products available from The MathWorks, Inc. The Simulink® software package includes a number of different tools, such as special user interfaces and navigational tools, e.g., a library browser, which will be referenced in the description to follow. Further details of these tools and interfaces can be had with reference to available Simulink® and MATLAB® product documentation. It will be understood, however, that other block diagram based modeling software platforms could be used.

The term "graphical block diagram" refers to a set of graphical blocks (or nodes) and

a set of lines (or signals) that carry data between the graphical blocks. Each graphical block typically performs a function and that function (or equation) is a sub-component of an overall set of equations describing a dynamic system. The function may be mathematical in nature or it may be an operation such as reading data from a hardware device. The graphical blocks may be parameterized with user-defined values, as will be described.

Using the functions or equations defined by each of the blocks, the graphical block diagrams can be executed to produce simulation results and generate textual software or firmware source code automatically as defined by the graphical blocks and signals in a model. Each of the equations is defined in a corresponding method (code module). For example, an output method, when invoked by the simulator 50 during model execution, determines an output signal y based on a given input signal u and block parameter value(s) p.

The module 40 enables users to copy graphical blocks into their models from the libraries 44 (or, optionally, from external libraries). Alternatively, or in addition, the module 40 allows a user to implement a custom block for use in a model and to add that custom block to the library 44 if the user so chooses. In a Simulink® product context, such custom blocks are referred to as "S-function" blocks.

FIGS. 3A-B illustrate examples of block diagrams created by the module 40. The block diagrams of FIGS. 3A and 3B implement the same functionality as the blocks diagrams depicted in FIGS. 1A and 1B, respectively, but use the blocks 52, 54 (from FIG. 2), collectively, to provide interpolation functionality in a more efficient manner, when the breakpoints of tables 20 are in common for the first input ports and the breakpoint sets for the second input ports are also in common with each other.

Referring to FIG. 3A, a first block diagram 60 includes the pre-lookup index search block 52. The block 52 receives an input value ("u_1") from a single input port ("In1") 64 and provides output values ("k" and "f", as later defined) to the interpolation block 54, which in turn provides an output value ("y_1") to an output port ("Out1") 66. Referring to FIG. 3B, a second block diagram 70 includes two of the pre-lookup index search blocks 52, indicated as 52a and 52b, and four of the interpolation blocks 54, indicated as 54a, 54b, 54c, 54d. The block 52a receives a first input value ("u_1") from a first input port ("In1") 72a and provides outputs k and f to each of the blocks 54a-54d. The block 52b receives a second input value ("u_2") from a second input port ("In2") 72b, and provides outputs k, f, distinct

from the k,f of block 52a, to each of the blocks 54a-54d. Each of the blocks 54a-54d provides a corresponding output y, indicated as y1, y2, y3, y4, respectively, to respective, corresponding output ports ("Out1") 76a, ("Out2") 76b, ("Out3") 76c, ("Out4") 76d, respectively. The blocks 52 and 54 can be implemented to perform any type of index search operation and interpolation (including extrapolation) algorithm, respectively.

For explicit pre-lookup index search and interpolation using pre-lookup block diagram constructs, the inputs to the module 40 include the following: i) input values (or table input values), connected graphically to input ports of the pre-lookup blocks 52; ii) breakpoint data set values for the pre-lookup blocks 52; and iii) table data values for the interpolation blocks 54. The module 40 enables a user to construct a block diagram to graphically connect the table input values to the pre-lookup index search blocks 52, which in turn are semantically connected to the interpolation blocks 54. Typically, to be useful, the outputs of the interpolation blocks 54 are graphically connected to other blocks in the block diagram so that the result of the interpolation calculation is used.

The operation of the models in FIGS. 3A and 3B will now be described in more detail. Each pre-lookup index search block 52, which performs an index search operation, has associated with it a "breakpoint data array", an ordered list of values of length N, {bp(0), bp(1), ..., bp(N-1)}. These breakpoint values define a set of regions that include regions corresponding to a set of N-1 intervals as well as regions below the first and above the last interval. Each block 52 uses its input value to compute as its outputs the numbers "k" and "f", as shown in FIGS. 3A and 3B. The number "k" is an interval index integer and the number "f" is a distance fraction "f". The value of "k" specifies in which region the input value resides. The block performs an index search to determine the value of k. Using a linear index search, the blocks determines k for an input value by checking each consecutive interval i, where i = 0, 1, 2..., N-1, to determine if the input value resides in that interval. That is, for an input value "u", the block determines if $bp(0) \leq u < bp(1)$ for a first interval (i = 0), $bp(1) \leq u < bp(2)$ for a second interval (i = 1), and so forth, that is, testing each consecutive interval i from 0 through i = N-1, until the block finds an inequality for a given interval to be true. The value i for the interval for which the inequality is determined to be true is the value k. Although a linear search algorithm has been described, any type of index search algorithm can be used to determine the value of k. For input values that are within the

first through the last interval in the breakpoint data, the value of "f" specifies the normalized distance on the interval specified by k (that is, $0.0 \leq f < 1.0$). If an input value falls within regions outside of the first through the next-to-last intervals, then an extrapolation is performed and the distance fraction for the extrapolation is:

$$u < bp(0): \; k = 0; \; f = (u - bp(0)) / (bp(1) - bp(0)) \qquad \text{Eq. 1}$$

$$u >= bp(N-1): \; k = N-2; \; f = (u - bp(N-2)) / (bp(N-1) - bp(N-2)) \qquad \text{Eq. 2}$$

Thus, the pre-lookup index search block locates an input value's relative position within a range of numbers (that is, the breakpoints in the breakpoint data set) and returns an array of the interval index "k" and distance fraction "f" that the input value reaches into the kth interval. The pre-lookup index search results (k, f) are in turn inputs to one or more ports on the interpolation blocks 54, which perform the desired interpolation.

For example, and as illustrated in FIG. 3A, a 1-Dimensional (1-D) linear interpolation y, using pre-lookup block output (k,f) and N Table values {T(0), T(1), …, T(N-1)} corresponding to the breakpoint data set { bp(0), bp(1), …, bp(N-1)} is:

$$y = T(k) + f*(T(k+1) - T(k)) \qquad \text{Eq. 3}$$

The interpolation calculation of Eq. 3 is well known among table look-up programmers and is also used in prior art look-up block implementations.

Therefore, in its most basic form, the module 40 uses the pre-lookup and interpolation blocks 52, 54 to split the operation of a conventional one-dimensional table look-up (one input value and one output value) into the index search portion and the interpolation portion of the calculation, as shown in FIG. 3A. Referring back to FIG. 1A, the block 12 performs the calculation for the index search and the interpolation internally.

For a multi-dimensional example, FIG. 1B shows a traditional lookup that perform 8 index searches and 4 interpolations, while FIG. 3B shows an equivalent construct that performs 2 index searches and 4 interpolations.

Referring to FIG. 4, assume that the data used the two-dimensional example shown in FIG. 3B is as follows:

Breakpoints for 1st dimension: [10, 20, 20]

Breakpoints for 2nd dimension: [10, 20, 30]

Table data for all tables: [4 5 6; 16 19 20; 10 18 23] (semicolon indicates end of row)

For simplicity of the example, the table data is all the same, but it could just as easily be different for each table with no impact on the number of operations used in the interpolation step. Likewise, the breakpoint data sets for the different dimensions need not be the same. Further assume that the input value u_1 of input port In1 is 12 and the input value u_2 of input port In2 is 28. The index search operation for the 1st dimension ("search 1") uses u_1 and the 1st dimension breakpoints to determine search 1 outputs (k, f). The index search operation for the 2nd dimension (search 2) uses u_2 and the 2nd dimension breakpoints to determine search 2 results (k, f). For search 1, the value of index "k" is 0. That is, the input value 12 resides in interval 0 (10 to 20). For search 2, the value of index "k" is 1, as the input value 28 resides in interval 1 (20 to 30). The distance fraction "f" for each of the two dimensions is as follows:

<u>1st Dimension:</u>

Distance fraction (f), Interval 0 = [(12-10)/20-10)] = 0.2

<u>2nd Dimension:</u>

Distance fraction (f), Interval 1 = [(28-20)/(30-20)] = 0.8

Thus, the output of the pre-lookup index search block 52a (search 1) is (0, 0.2). The output of the pre-lookup index search block 52b (search 2) is (1, 0.8).

The interpolation calculations performed by each of the interpolation blocks 54a-54d are as follows:

<u>1st Dimension:</u>

$y_{1d\_1} = 5 + 0.2(19-5) = 7.8$

$y_{1d\_2} = 6 + 0.2(20-6) = 8.8$

<u>2nd Dimension:</u>

$y_{2d} = 7.8 + 0.8(8.8-7.8) = 8.6$

Therefore, in the example of FIG. 3B, the value of each of the outputs y_1, y_2, y_3, y_4 of the interpolation blocks 54a, 54b, 54c, 54d, respectively, is 8.6

In the described embodiment, and as noted earlier, the blocks 52, 54 reside in the library 44 and so can be used by multiple users in their respective models. A user selects the blocks using a menu provided by a library browser. Having opened a model window for a model to be generated and/or edited, the user copies the selected blocks from the library window to the model window, e.g., by selecting ("clicking on") a corresponding library node or icon, dragging the block from the library browser and dropping it in the model window.

Referring now to FIG. 5, a screen display of a model window 90 shows the placement of the selected blocks 52a, 52b and 54 within a user's model 92. In the exemplary block diagram model 92 shown, the user has also included the constant ports or blocks 72a, 72b to supply the table input values. In keeping with the previous example, these values are 12 and 28. Only one of the four blocks 54 in the previous example is shown in the model window. To further illustrate the differences between the conventional lookup block 20 (from FIG. 1B) and the new blocks 52a, 52b and 54, as well as to highlight the fact that the module 40 can (optionally) support the conventional style lookup block 20 as well, the model 92 also includes the lookup block 20.

Further included in the model 82 are display blocks 94a, 94b, 94c and 94d for displaying the outputs of the blocks 20, 52a, 52b and 54, respectively. The model 92 also includes connection lines 96a-96j. Lines 96a and 96b connect the constant 72a and the constant 72b, respectively, to the lookup block 20, which is connected to the display 94a by the line 96e. Lines 96c and 96d connect the constant 72a and the constant 72b to the pre-lookup blocks 52a, 52b, respectively. The block 52a is connected to the display 94b by the line 96b and to the interpolation block 54 by the line 96f. The block 52b is connected to the display 94c by the line 96i and to the interpolation block 54 by the line 96g. The block 54 is connected to the display 94d by the line 96j. During model execution, as shown, the displays 94a through 94d display the appropriate output values for the blocks to which they are connected, given the same data values used in the example described earlier.

The behavior of a block in a block library may be defined in terms of "top level"

parameters that the user may specify when creating an instance of the block. The behavior of the block is akin to the behavior of a template class as used in object oriented programming languages such as C++. Instances associated with the graphical class in the library 44 have parameter specification interfaces that allow a user to set values for these "top-level" parameters. On the user's GUI, such as the GUI 18 (of FIG. 2), such parameter specification interfaces take the form of a dialog box with various parameter fields. A user can therefore change the top-level parameters of an instance by changing parameter data in a parameter field in the dialog box for that instance. Alternatively, a user could specify the parameters programmatically via a textual API.

Referring to FIG. 6, a parameters dialog box 100 for the (prior art) interpolation lookup block 20 is shown. The user uses parameter fields 102, 104 and 106 to set respective parameter values, or, more specifically, 1st dimension (row) break point data set values 102, 2nd dimension (column) break point data set values 104, and table data 106, respectively.

FIGS. 7A and 7B illustrate dialog boxes 110a and 110b for the blocks 52a and 52b, respectively. Each of the boxes 110 includes a parameter field 112 for specifying the breakpoint data array for the search/dimension to which the block corresponds. The field 112 in the box 110a specifies the row breakpoint data for block 52a and the field 112 in the box 110b specifies the column breakpoint data for block 52b. The boxes 110 further include a field 114 for specifying an index search algorithm, e.g., linear search, to be used, as well as fields 115 for specifying extrapolation information for handling cases in which the input values fall outside the breakpoint data defined intervals as described earlier.

Referring to FIG. 8, a parameters dialog box 116 for the interpolation block 54 is shown. It includes a table data field 118 for specifying the table data.

FIG. 9 shows the selection of the block 54 using a debugger tool

Referring to FIG. 10, the window of the debugger tool 120 displays the input and output values for the selected block (as well as other previously selected blocks) while the model is executing.

Thus, the user determines the table input values, and graphically connects signals from anywhere in the block diagram to N pre-lookup blocks, which in turn are connected to an interpolation block set for N dimensions to perform an n-dimensional interpolation. The user also determines the breakpoint and table data, which the user provides to the model by

entering the data in the appropriate dialog boxes. The order in which the blocks are added and connected can vary, as long as the resulting block diagram is functionally the same.

As illustrated in FIG. 6, the prior interpolation lookup requires that the user specify index search parameters, more specifically, breakpoint data, for each interpolation lookup block that uses that data. If several different interpolation lookup blocks (with the same or different table data) use the same breakpoint set(s), a user must specify the breakpoint data for each of those blocks individually. Such an approach is time-consuming and error-prone. As shown in FIGS. 7A-7B, using the two step approach (that is, separate blocks for index search and interpolated lookup), the breakpoint data is specified by the user in a centralized manner, that is, using n pre-lookup index search blocks for n-dimensions, thus ensuring that all interpolation blocks (lookup tables) use the same breakpoint data.

In Simulink®, blocks can be graphically connected using sequences of lines, named goto/from block pairs, mux/demux block pairs, bus creator/bus selector block pairs, subsystem input ports, subsystem output ports, and data store memory blocks. Other block diagram tools may have other graphical connection semantics, but that does not limit how the pre-lookup index search and interpolation (using pre-lookup index search) blocks operate.

In the embodiment described above, and referring back to FIG. 1, the compiler 46 interprets a model for which a two step interpolation has been explicitly specified and drawn (that is, two separate blocks for index search and interpolated lookup are used). As an alternative embodiment to the explicit two step approach described thus far, the model built by the model constructor/editor 42 implements a block diagram that uses prior art interpolation lookup blocks (such as the blocks 20 in FIG. 1B) and the compiler 48 interprets the blocks as if they had been drawn as separate blocks for index search and interpolated lookup by detecting that the blocks 20 share common input values and breakpoint data sets. That is, the compiler 48 interprets some number "*m*" of interpolation lookup blocks (such as blocks 20) with common input values and breakpoint data set(s) as being a two step interpolation configuration of blocks, having index search blocks (such as blocks 52, as shown in FIG. 3B), one for each of the shared inputs, connected to and feeding *m* interpolation blocks (such as blocks 54, also shown in FIG. 3B).

It is to be understood that while the invention has been described in conjunction with the detailed description thereof, the foregoing description is intended to illustrate and not

limit the scope of the invention, which is defined by the scope of the appended claims. Other embodiments are within the scope of the following claims.

What is claimed is:

1